功能：计算生成平面C2连续 样条闭曲线数据点处的一阶导矢。
输入参数：m_aVertex-数据点，点数n隐含其中；m_aV[i]-张力参数；数据点参数化方法确定节点区间长度m_delta[i]，均为受保护成员。
输出参数：m_Cx，m_Cy-数据点处的一阶导矢x、y分量。

```
Void InitCycleNuSpline()
{
        CArray<CArray<double,double>,CArray<double,double>&>  U;
        CArray<double,double> tx;
        CArray<double,double> ty;
        CArray<double,double> a;
        CArray<double,double> b;
        CArray<double,double> c;
        CArray<double,double> dx;
        CArray<double,double> dy;
        int n=m_aVertex.GetSize();
        U.SetSize(n);
        for(int i=0;i<n;i++)    U[i].SetSize(n);
        m_Cx.RemoveAll();
        m_Cy.RemoveAll();
        m_Cx.SetSize(n);
        m_Cy.SetSize(n);
        a.SetSize(n);
        b.SetSize(n);
        c.SetSize(n);
        dx.SetSize(n);
        dy.SetSize(n);
        for(int ii=0;ii<n;ii++)
        {
                tx.Add(double(m_aVertex[ii].x));
                ty.Add(double(m_aVertex[ii].y));
        }

        for(i=1;i<n-1;i++)
        {
                a[i]=m_delta[i];
                b[i]=2*m_delta[i-1]+2*m_delta[i]+0.5*m_delta[i-1]*m_delta[i]*m_aV[i];
                c[i]=m_delta[i-1];

dx[i]=3.*(m_delta[i]*(tx[i]-tx[i-1])/m_delta[i-1]+m_delta[i-1]*(tx[i+1]-tx[i])/m_delta[i]);

dy[i]=3.*(m_delta[i]*(ty[i]-ty[i-1])/m_delta[i-1]+m_delta[i-1]*(ty[i+1]-ty[i])/m_delta[i]);
        }
        a[n-1]=m_delta[0];
        b[n-1]=2*m_delta[n-2]+2*m_delta[0]+0.5*m_delta[n-2]*m_delta[0]*m_aV[n-1];
        c[n-1]=m_delta[n-2];
        dx[n-1]=3.*(m_delta[0]/m_delta[n-2]*(tx[n-1]-tx[n-2])+m_delta[n-2]/m_delta[0]*(tx[1]-tx[0]));
        dy[n-1]=3.*(m_delta[0]/m_delta[n-2]*(ty[n-1]-ty[n-2])+m_delta[n-2]/m_delta[0]*(ty[1]-ty[0]));
        for(i=0;i<n;i++)
                for(int j=0;j<n;j++)
                        U[i][j]=0.;
        for(i=1;i<n-1;i++)
        {
                U[i][i]=b[i];
                U[i][i+1]=c[i];
                U[i][i-1]=a[i];
        }
        U[0][n-1]=-1;
        U[0][0]=1;
        U[n-1][1]=c[n-1];
        U[n-1][n-1]=b[n-1];
        U[n-1][n-2]=a[n-1];
        dx[0]=0.;
        dy[0]=0.;
        int chl=0;
        for(int  tt=0;tt<n-1;tt++)
        {
                double  d=U[tt][tt];
                bool  n_move=false;
                for(int j=tt+1;j<n;j++)
                {
                        if(fabs(d)<fabs(U[j][tt]))
                        {
                                d=U[j][tt];
                                chl=j;
                                n_move=true;
```

```
                }
        }
        double  tem1,tem2,tmpx,tmpy;
        if(n_move)
        {
                for(int mm=tt;mm<n;mm++)
                {
                        tem1=U[tt][mm];
                        tem2=U[chl][mm];
                        U[tt][mm]=tem2;
                        U[chl][mm]=tem1;
                }
                tmpx=dx[tt],tmpy=dy[tt];
                dx[tt]=dx[chl],dy[tt]=dy[chl];
                dx[chl]=tmpx,dy[chl]=tmpy;
        }
        for(j=tt+1;j<n;j++)
        {
                double  l=U[j][tt]/U[tt][tt];
                dx[j]=dx[j]-l*dx[tt];
                dy[j]=dy[j]-l*dy[tt];
                for(int v=0;v<n;v++) U[j][v]=U[j][v]-l*U[tt][v];
        }
    }
    m_Cx[n-1]=dx[n-1]/U[n-1][n-1];
    m_Cy[n-1]=dy[n-1]/U[n-1][n-1];
    for(int ll=n-2;ll>=0;ll--)
    {
        double  xx=0.,yy=0.;
        for(int k=ll+1;k<n;k++)
        {
                xx=xx+U[ll][k]*m_Cx[k];
                yy=yy+U[ll][k]*m_Cy[k];
        }
        m_Cx[ll]=(dx[ll]-xx)/U[ll][ll];
        m_Cy[ll]=(dy[ll]-yy)/U[ll][ll];
    }
}
```